

Kontrollstrukturen

Hochsprachen bieten im eine Vielzahl verschiedener Konstrukte zur bedingten und wiederholten Ausführung von Anweisungen. Zum Beispiel:

- `if (...) { ... } else { ... }`
- `switch (...) { case ...: ... case ...: ... default: ... }`
- `while { ... }`
- `do { ... } while;`
- `for (...; ...; ...) { ... }`
- `goto ...; ... label: ...`
- ...

Kontrollstrukturen

Kontrollstrukturen lassen sich auf einfachere zurückführen. Beispiel:

```
for (i = 0; i < 16; i++) { anw(); }
```

erster Vereinfachungsschritt:

```
i = 0;
while (i < 16) {
    anw();
    i++;
}
```

nächste Vereinfachung:

```
i = 0;
goto next;
loop: anw();
    i++;
next: if (i < 16) goto loop;
```

Kontrollstrukturen

Kompliziertere Bedingungen lassen sich aufteilen. Beispiel:

```
if (a > b && a != 0) { anw(); }
```

erster Vereinfachungsschritt:

```
if (! (a > b)) goto end;  
if (! (a != 0)) goto end;  
anw();  
end: ;
```

nächste Vereinfachung:

```
if (a <= b) goto end;  
if (a == 0) goto end;  
anw();  
end: ;
```

Kontrollstrukturen

Bedingte Ausführungen von Anweisungen sowie Schleifenkonstrukte lassen sich auf ein oder mehrere

```
goto label;  
if (a op b) goto label;      op ∈ {<, >, =, !=, <=, >=, ...}
```

Anweisungen abbilden.

ein unbedingten Sprung (goto label) in Assembler-Code:

```
jmp address
```

ein bedingter Sprung (if (a op b) goto label) in Assembler-Code:

```
jop a, b, address          <- Problem: drei Operanden notwendig
```

```
cmp a, b  
jop address                daher häufig implizites Register:  
                           Condition-Code-/Flags-Register
```

Kontrollstrukturen (bedingte Sprünge)

Beispiel:

```
char a, b;  
if (a < b) goto label;
```

ergibt kompiliert für i80x86:

```
cmp b, a  
jl address          /* "jump if less" */
```

ergibt kompiliert für m68x05:

```
lda a  
cmp b  
blo address         /* branch if lower */
```

Kontrollstrukturen (bedingte Sprünge)

mögliche Bedingungen:

	i80x80	m68x05
=	je	beq
!=	jne	bne
>, >= (unsigned)	ja, jae	bhi, bhs
<, <= (unsigned)	jb, jbe	blo, bls
>, >= (signed)	jg, jge	bpl, -
<, <= (signed)	jl, jle	bmi, -
...		

Kontrollstrukturen (bedingte Sprünge)

Beispiel (i80x86):

```
register short int a; /* liegt in Register %ebx */
short int b; /* hat Adresse 124 */
if (a < b && a != 0) then goto label;
/* label entspricht Adresse 25 */
```

ergibt kompiliert:

```
09:    cmpw 124, %bx
10:    jge 13;
11:    cmpw %bx, $0
12:    jne 25
13:    ...
...
25:    ...
...
```

Kontrollstrukturen (Hinweise)

Auch viele Arithmetik-Befehle setzen die Condition-Codes (CC) im Flags-Register (Beispiele):

- Zero-Flag: gesetzt, wenn Ergebnis 0 ist
- Negative-Flag: gesetzt, wenn das höchstwertige Bit des Ergebnisses 1 ist
- Carry-Flag: bei Übertrag gesetzt
- ...

Bedingte Sprünge benutzen die Condition-Codes (Beispiel: i80x86):

- je („jump if equal“) verzweigt, wenn das Zero-Bit gesetzt ist
- jl („jump if less“) verzweigt, wenn das Sign-Bit gesetzt ist
- jle („jump if less or equal“) verzweigt, wenn das Sign- oder Zero-Bit gesetzt sind
- ...

cmp-Befehl subtrahiert die zwei Operanden, setzt die CC-Bits entsprechend, speichert aber das Subtraktionsergebnis nicht.

Kontrollstrukturen (Hinweise)

Statt einer Adresse als Sprungziel wird häufig die Differenz zur Adresse der aktuellen (oder nächsten) Instruktion angegeben (meist eine kleinere Zahl). Beispiel:

08: ...	08: ...
09: jle 25	09: jle +16
10: ...	10: ...

Vorteil:

- weniger Speicheraufwand
- schnellere Befehlsausführung
- unabhängig von der tatsächlichen Lage des Programms im Speicher

Nachteil:

- mühsamere Berechnung (wird meist vom Assembler übernommen)
- nicht alle Sprungziele erreichbar (lange Version zusätzlich notwendig)